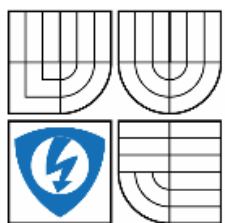




VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



**FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH
TECHNOLOGIÍ**
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AC DRIVES MODELING IN MODELICA

MODELOVÁNÍ STŘÍDAVÝCH POHONŮ V JAZYCE MODELICA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

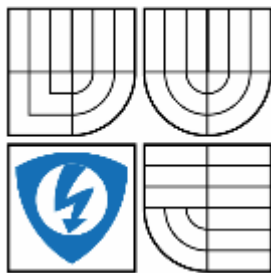
AUTHOR
AUTOR PRÁCE

GORAN LALOVIĆ

SUPERVISOR
VEDOUCÍ PRÁCE

doc. Ing. PAVEL VÁCLAVEK, Ph.D.

BRNO 2014



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Goran Lalović
Ročník: 3

ID: 146049
Akademický rok: 2013/2014

NÁZEV TÉMATU:

Modelování střídavých pohonů v jazyce Modelica

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je příprava modelů střídavých pohonů v prostředí OpenModelica. Zadání lze shrnout do následujících bodů:

1. Zpracujte rešerši týkající se modelování synchronního a asynchronního motoru a modelovacích nástrojů.
2. V prostředí OpenModelica připravte modely synchronního a asynchronního motoru.
3. Porovnejte chování navržených modelů s dostupnými modely v prostředí Matlab-Simulink.
4. Pokuste se navrhnout SW nástroj umožňující zahrnutí navržených modelů v prostředí OpenModelica jako do simulačních schémat v prostředí Matlab-Simulink.
5. Ověřte a vyhodnotte vlastnosti navrženého řešení.

DOPORUČENÁ LITERATURA:

[1] Caha, Z.; Černý, M.: Elektrické pohony, Praha, SNTL 1990.
další dle pokynů vedoucího práce

Termín zadání: 10.2.2014

Termín odevzdání: 26.5.2014

Vedoucí práce: doc. Ing. Pavel Václavek, Ph.D.

Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRACT

This bachelor final project deals with the procedures of physical modeling and simulation of AC drives (electrical machines) in Modelica programming language and environment OpenModelica. First, there is an overview about the general characteristics of the models for basic types of AC machines, Asynchronous and Synchronous Machines, and controller structures in OpenModelica. Then, this bachelor's thesis concentrates on a behavior comparison of the already existing models in Matlab-Simulink and OpenModelica. The goal of this bachelor's thesis was making the FMU, translating, simulating and working with models from OpenModelica to Matlab-Simulink.

The theoretical description of the electrical machines is mainly based in the first two chapters. This bachelor final project includes the same modeling types of the electrical machines in different modeling languages, OpenModelica and Matlab-Simulink, and compares the simulations results. At the end, it gives some basic information about FMI standard, FMU and talks about translating models from OpenModelica, through JModelica to Matlab-Simulink.

KEYWORDS

AC Electrical Machines, Asynchronous Machines, Synchronous Machines, Modeling and Simulation, OpenModelica, Matlab-Simulink, JModelica.org, FMI, FMU, FMU Toolbox, Parameterization Definer

ABSTRAKT

Tato závěrečná bakalářská práce se zabývá s postupy fyzikálního modelování a simulaci střídavých pohonů v jazyce Modelica a prostředí OpenModelica. Za prvé, zde se nachází přehled všeobecných vlastností modelů pro základní typy střídavých pohonů, asynchronní a synchronní stroje a příslušných regulačních struktur v prostředí OpenModelica. Poté se bakalářská práce zaměřuje na srovnání chování vytvořených modelů s dostupnými modely v prostředí Matlab-Simulink. Cílem bakalářské práce bylo vyexportovat FMU, přeložit, odsimulovat a pracovat s modely z prostředí OpenModelica do Matlab-Simulink.

Teoretický popis základních typů střídavých pohonů se nachází na prvních dvou kapitolách. Ve třetí kapitole se bakalářská práce zabývá srovnáváním modelů vytvořených v OpenModelica a Matlab-Simulink, a porovnává výsledky simulace. Na konci, jsou uvedeny základní informace o standardu FMI, FMU a mluví se o překladu modely z prostředí OpenModelica, pomocí JModelica-y do Matlab-Simulink-u.

KLÍČOVÁ SLOVA

Střídavé pohony, Asynchronní Motor, Synchronní Motor, Modelování a Simulace, OpenModelica, Matlab-Simulink, Jmodelica.org, FMI, FMU, FMU Toolbox, Parameterization Definer

BIBLIOGRAPHIS CITATION:

LALOVIĆ, G. *AC drives modeling in OpenModelica*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, 2014. 47 pages. Supervisor doc. Ing. Pavel Václavek, Ph.D..

BIBLIOGRAFICKÁ CITACE MÉ PRÁCE:

LALOVIĆ, G. *Modelování střídavých pohonů v jazyce Modelica*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 47 s. Vedoucí bakalářské práce doc. Ing. Pavel Václavek, Ph.D..

PROHLÁŠENÍ

„Prohlašuji, že svou bakalářskou práci na téma *Modelování střídavých pohonů v jazyce Modelica* jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 25. května 2014

.....
Podpis autora

ACKNOWLEDGEMENT

First and foremost, the author wishes to express his gratitude to his supervisor of this bachelor final project, doc. Ing. Pavel Václavek, Ph.D., for the valuable guidance and advice. He was abundantly helpful and offered invaluable assistance, support and guidance. His willingness to motivate me contributed tremendously to my project.

I would also like to thank the authority of Brno University of Technology (BUT) for providing with a good environment and facilities to complete this project.

Finally, an honorable mention goes to my family and friends for their understandings and supports on me in completing this project. Without the help of the particular people that mentioned above, I would face many difficulties while doing this.

Table of Contents

1	INTRODUCTION	1
2	ELECTRICAL MACHINES	2
2.1	History of AC induction machine	2
2.2	Basic construction of AC induction machine	3
2.2.1	Stator	3
2.2.2	Rotor.....	3
2.3	Speed of an AC induction machine.....	4
2.4	Asynchronous machine	4
2.4.1	Mathematical description of asynchronous induction motors	5
2.5	Synchronous machine	7
2.6	Permanent Magnet Synchronous Machine (PMSM)	7
2.7	Mathematical description of synchronous motors	10
3	OPENMODELICA.....	12
3.1	Asynchronous motor in OpenModelica	12
3.1.1	Existing model of asynchronous induction machine with squirrel cage or ring rotor.....	12
3.1.2	Class parameterization modeling of AIMC	14
3.2	Synchronous motor in OpenModelica	18
3.2.1	Existing model of synchronous motor with permanent magnet	19
3.2.2	Class parameterization modeling of SM with permanent magnet and losses	19
3.2.3	PMSM with load	24
3.3	Sub models in OpenModelica	25
3.4	Instantiate model and text view	26
3.5	AIMC and PMSM documentation in OpenModelica	27
3.6	Record and variable definition in OME.....	27
4	FUNCTIONAL MOCK-UP INTERFACE – FMI	28
4.1	JModelica used for the compilation of the FMU	28
4.2	FMU Toolbox	29
4.3	Parameterization Definer	31
5	CONCLUSION.....	33

6	List of Figures.....	34
7	Bibliography	35
8	List of abbreviations and symbols	37
9	List of attachments.....	38

1 INTRODUCTION

No modern home should be without a few induction motors. Induction motors are everywhere and can be found in fridges, fans, vacuum cleaners, washing machines, dishwashers etc. The reason for high usage of induction machines is their low initial cost, long life, low maintenance requirements (minimal service requirements), high efficiency, automated control, they require no fuel, engine oil maintenance, battery service, and do not freeze in sub-zero temperatures.

Today, modeling and simulation are so important. It is a way for getting information about how something will behave without actually testing it in real life. OpenModelica is Modelica-based environment for making models and simulations of systems in real life.

This bachelor's thesis is divided into five important chapters:

- Electrical Machines;
- Modeling and Simulation;
- Verifying the correctness of simulation results;
- FMU export and making the software for model exchange;
- Conclusion.

Our aim in the first part of bachelor final project is to describe informally and theoretically induction machines, their parts – stator and rotor, basic Asynchronous and Synchronous Machines, how they work and what is the biggest difference between them.

Then, this bachelor's thesis deals with the modeling and simulation on PC in OpenModelica, one of the most promising modeling and simulation environments. We'll speak about basic characteristics of OpenModelica, making models and simulations of asynchronous induction and synchronous machines.

To make sure that models are correct, fourth chapter compares the simulation results of angular velocity and stator's currents from OpenModelica with the already existed models in Matlab-Simulink.

Purpose and aim of this final year project was to translate models from OpenModelica to Matlab-Simulink by using the FMI standardized interface. Task was to make the software that can load an FMU of AIMC or PMSM and that can change the FMU parameters of the above mentioned induction machines.

At the end, there are a few words about the whole project, FMI standard, conclusion and advantages of using FMU Toolbox with Parameterization Definer.

2 ELECTRICAL MACHINES

All electrical machines convert one form of energy to another. Because of that they are separated into three different categories [1]:

- Generator Machines - which convert mechanical to electrical energy;
- Motors - which convert electrical to mechanical energy;
- Transformers - this changes the voltage level of an AC.

Our aim in this bachelor's thesis is the alternating current (AC) drives or motors and their characteristics. The main task of the alternating current drives is to create the motion. In order to successfully accomplish this task, electrical machines, motors, convert electrical energy to mechanical energy. Mechanical energy is used for, for example, rotation and linear movement, in compressors, lifting subjects, pumps etc.

AC induction motors are the most common motors used in industry. The most important advantages of AC induction motors are simple design, direct connection to an AC power station and low-cost.

2.1 History of AC induction machine

According to (Tesla Constructs the First Induction Motor., 2005), in 1882, famous Serbian scientist, Nikola Tesla, discovered the rotating magnetic field, a fundamental principle in physics and one of the greatest discoveries of all times. In February, 1882, Nikola Tesla was walking with his friend through a city park in Budapest, Hungary and he was reciting stanzas from Goethe's Faust. Suddenly, the elusive solution to the rotating magnetic field, which he had been seeking for a long time, flashed through his mind and in his mind he saw clearly an iron rotor spinning rapidly in an rotating magnetic field, produced by the interaction of two alternating currents out of step with each other. This glorious moment was one of the ten greatest discoveries of all times. Tesla was gifted with the intense power of visualization and exceptional memory from early youth on. He was able to fully find, construct, develop and perfect his inventions completely in his mind before committing them to paper. From his memory he constructed the first induction AC machine - motor. In the summer of 1883, Tesla was working in Strasburg, France, where he built his first alternating current induction motor model and saw it run. Today, Tesla's AC induction machine is widely used everywhere throughout the world, in industry and household appliances. It started the Industrial Revolution at the turn of the century[2].

2.2 Basic construction of AC induction machine

Like many other electrical machines, AC induction machine has two basic parts:

- Fixed “stator”;
- Spinning “rotor” with a carefully dimensioned air gap.

2.2.1 Stator

The stator has coils supplied with AC to produce a rotating magnetic field. A three-phase alternating current induction motor is specific because it creates rotating magnetic field naturally in the stator. First important part of AC motor, the stator, is made of several thin laminations. Laminations are clamped together to stator core. The number of poles of an AC induction motor depends on the internal connection of the stator windings. The stator windings are connected directly to the power source [3].

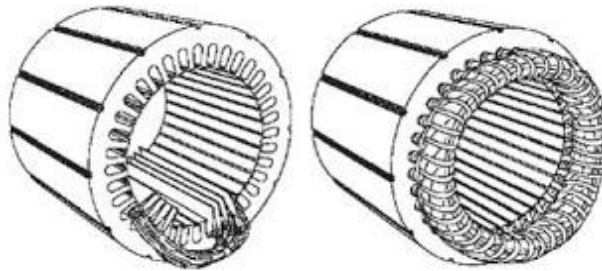


Figure 1: Stator [4]

2.2.2 Rotor

The rotor, which spins inside, is made up of several thin laminations, as well. There are two basic types of rotor:

- Squirrel cage rotors - because it has a simple and rugged construction, 90% of AC induction machines have this;
- Slip ring rotors.

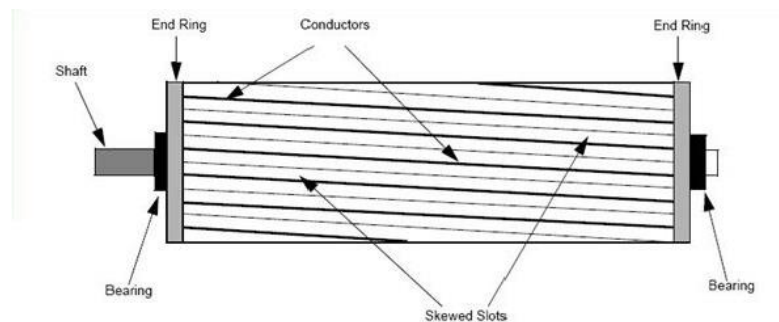


Figure 2: Squirrel cage rotor [5]

2.3 Speed of an AC induction machine

Synchronous speed of the magnetic field created in the stator rotates at:

$$N_s = 60 \frac{f}{p} \text{ [RPM]} \quad (2.1)$$

Where:

N_s	synchronous speed of the stator magnetic field
f	Frequency in Hertz
p	Number of pole pairs

The magnetic field produced in the rotor is alternating in nature. Rotor runs in the same direction as the flux that is produced in stator. Rotor tries to “catch up” with the rotating flux but never succeeds in that task. The reason for this is that rotor runs slower than a speed of stator field. This speed is called “Base Speed“ (N_B) and the difference between N_B and N_s (Synchronous speed of the stator magnetic field) is called the “slip” [3]:

$$slip = \frac{N_s - N_B}{N_s} 100 \text{ [%]} \quad (2.2)$$

2.4 Asynchronous machine

One of the most popular alternating current machines is asynchronous machine. Our goal in this, 2.4, chapter is to give a main theoretical description of the asynchronous machine.

The construction of asynchronous machine is very simple, rugged, reliable and economical. Asynchronous machine does not require a lot of maintenance and the main reason for this is that asynchronous machine doesn't have commutator, brushes and slip rings.

Three-phase or multi-phase alternating current in the stator (cylindrical shape is typically for the stator) produces a magnetic field. Squirrel cage or slip ring rotor is not connected to an external voltage source. The rotor of the slip ring motor has more winding turns than a squirrel cage rotor. That's why rotor of the slip ring machine produces the higher voltage level and the lower current, compared to a squirrel cage rotor. The construction of the squirrel cage rotor is cheaper and easier [6].

Rotor never rotates with the same speed (synchronization speed) as the stator's rotating field. As there must be a flux wave rotating relative to the rotor to provide torque, the rotor always runs at different speed (slightly lower or higher), asynchronous

speed, which gives the name for this machine. The asynchronous machine is also called induction machine [6].

2.4.1 Mathematical description of asynchronous induction motors

For the derivation of the mathematical model - asynchronous induction motor, we rely on some simplifying assumptions. We will assume symmetrical three-phase supply, harmonic voltage U_s with frequency ω .

$$\begin{aligned} u_a &= U_s \cos(\omega t), \\ u_b &= U_s \cos\left(\omega t - \frac{2}{3}\pi\right), \\ u_c &= U_s \cos\left(\omega t - \frac{4}{3}\pi\right). \end{aligned} \quad (2.3)$$

Resistances and inductances are same in various stages of stator and rotor windings; rotor and stator windings have the same number of turns and they are spatially symmetrically distributed into grooves. Furthermore, we assume a linear magnetization characteristic, harmonic distribution of magnetic induction in the air gap and negligible heat and iron losses.

$$\begin{aligned} \mathbf{u}_S^S &= U_{S\alpha} + j U_{S\beta} & \mathbf{i}_S^S &= i_{S\alpha} + j i_{S\beta} \\ \mathbf{u}_R^R &= 0 & \mathbf{i}_R^R &= i_{R\alpha} + j i_{R\beta} \end{aligned} \quad (2.4)$$

Where, superscript expresses the coordinate system with stator S and rotor R .

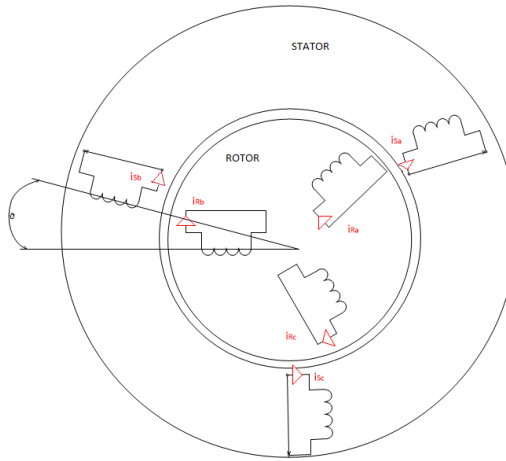


Figure 3: Winding for deriving the mathematical model of ASM

$$\begin{aligned} \mathbf{u}_S^S &= R_S \mathbf{i}_S^S + \frac{d}{dt} \boldsymbol{\psi}_S^S \\ \mathbf{u}_R^R &= R_R \mathbf{i}_R^R + \frac{d}{dt} \boldsymbol{\psi}_R^R = 0 \end{aligned} \quad (2.5)$$

R_S and R_R are the stator and rotor resistances. Stator and rotor inductance are L_S and L_R .

$$\begin{aligned}\psi_S^S &= L_S \mathbf{i}_S^S + L_m e^{j\theta} \mathbf{i}_R^R \\ \psi_R^R &= L_R \mathbf{i}_R^R + L_m e^{j\theta} \mathbf{i}_S^S\end{aligned}\quad (2.6)$$

Main field inductance in henries - L_m varies and depends on the angular position of the rotor.

Using the Park's transformation equations from above (2.6) can be expressed in a common coordinate system K . If the angle between the axis of the stator winding and the common coordinate winding is marked as ϕ^K then the coordinate system R is rotated by an angle $-(\phi - \phi^K)$. In this scenario the voltages and currents in the coordinate system K are:

$$\begin{aligned}\mathbf{u}_S^K &= \mathbf{u}_S^S e^{-j\phi^K}, \\ \mathbf{i}_S^K &= \mathbf{i}_S^S e^{-j\phi^K}, \\ \mathbf{i}_R^K &= \mathbf{i}_R^R e^{-j(\phi^K - \phi)}.\end{aligned}\quad (2.7)$$

Substituting equations (2.5) and (2.6) into equations (2.7) we obtain the equation in a common coordinate system K , from which it is seen that the coupled magnetic flow is independent of the relative position of the stator winding and rotor.

$$\begin{aligned}\mathbf{u}_S^K &= R_S \mathbf{i}_S^K + \frac{d}{dt} \psi_S^K + j\omega^K \psi_S^K \\ 0 &= R_R \mathbf{i}_R^K + \frac{d}{dt} \psi_R^K + j(\omega^K - \omega_e) \psi_R^K \\ \psi_S^K &= L_S \mathbf{i}_S^K + L_m \mathbf{i}_R^K \\ \psi_R^K &= L_S \mathbf{i}_R^K + L_m \mathbf{i}_S^K\end{aligned}\quad (2.8)$$

Torque is independent of the coordinate system. For subsequent simulation of the engine it is used the expression for the moment engine:

$$T_e = \frac{3}{2} P_p \operatorname{Im}\{\psi^K * \mathbf{i}^K\} = \frac{3}{2} P_p \operatorname{Im}\{\psi_S * \mathbf{i}_S\}.\quad (2.9)$$

The relationship between the speed of the rotor ω_m and the electrical angular velocity ω_e is:

$$\omega_e = P_p \omega_m.\quad (2.10)$$

The number of pole pairs of the motor is marked with the P_p . Slip, which is characterized by the difference between angular speed of the rotor and synchronous angular velocity of the magnetic field, can be mathematically expressed as:

$$S = 1 - \frac{\omega_m}{\omega_s}, \quad (2.11)$$

$$\omega_s = \frac{2\pi f}{P_p}. \quad (2.12)$$

In this case, mechanical angular velocity without any friction is:

$$\frac{d}{dt}\omega_m = \frac{1}{J} (T_e - T_L). \quad (2.13)$$

T_L is the load torque and J is the moment of inertia of the motor. By substituting expression (2.9) into equation (2.13) and the conversion of mechanical to electrical speed, electrical angular velocity can be expressed as:

$$\frac{d}{dt}\omega_e = \frac{P_p}{J} \left(\frac{3}{2} P_p \operatorname{Im}\{\Psi_s * \mathbf{i}_s\} - T_L \right). \quad (2.13)$$

2.5 Synchronous machine

A second most popular alternating current machine is synchronous machine.

The construction of synchronous machine is very similar to the asynchronous machine; they are very simple, rugged, reliable and economical. Stator has the same construction and the only difference is in the rotor construction. To create a continuous magnetic field, the connection to the rotor coils are taken out and connected to an external current source [6]. That's why rotor always runs synchronously to the stator's flux, which gives the name for this group of AC machines – synchronous machines.

2.6 Permanent Magnet Synchronous Machine (PMSM)

Special type of the synchronous machine is the Permanent Magnet Synchronous Machine (PMSM). Construction of the whole machine is changed and simplified. The rotor of PMSM contains permanent magnets which replace the connection of the rotor coils to an external power source [6]. New technology of alloys allows the production of permanent magnets. The most common materials for the permanent magnets are Neodymium-Boron Iron ($NdFeB$) and Samarium-Cobalt ($SmCo$). Magnetic flux for usual ferrite is around $0.3 - 0.4$ T, but the magnetic flux of special alloys is in the range of $0.8 - 1.2$ T.

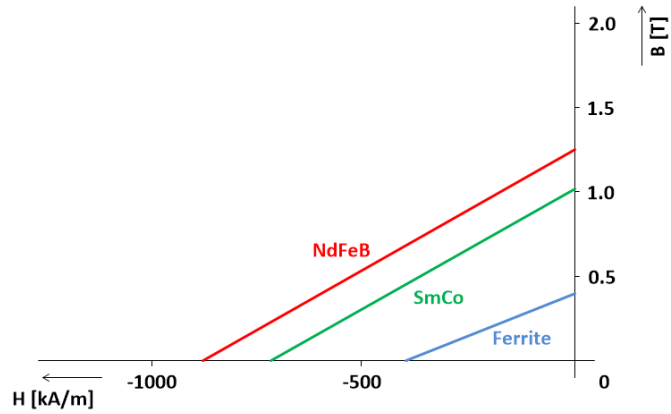


Figure 4: Flux Density (B) versus Magnetizing Field (H) of Permanent Magnetic Materials

There are a lot of advantages that PMSM has and the greatest is low volume of the PMSM in contrast with other types of motors. Reason for this is next - the construction of the rotor and whole machine is minimized and simplified; the weight of the PMSM is reduced and lighter; the compactness of the machine is enhanced. These factors play such important role for the design and construction of electric or hybrid road vehicles.

Lot of possible constructions for a PMSM are available. The common layout is an inner rotor and on the next figure (Figure 4) is shown PMSM with surface mounted magnets. Red means a North Magnetic Pole regarded from the stator and blue color means a South Magnetic Pole. Another possibility shows Figure 5 - PMSM with inner rotor and buried magnets.

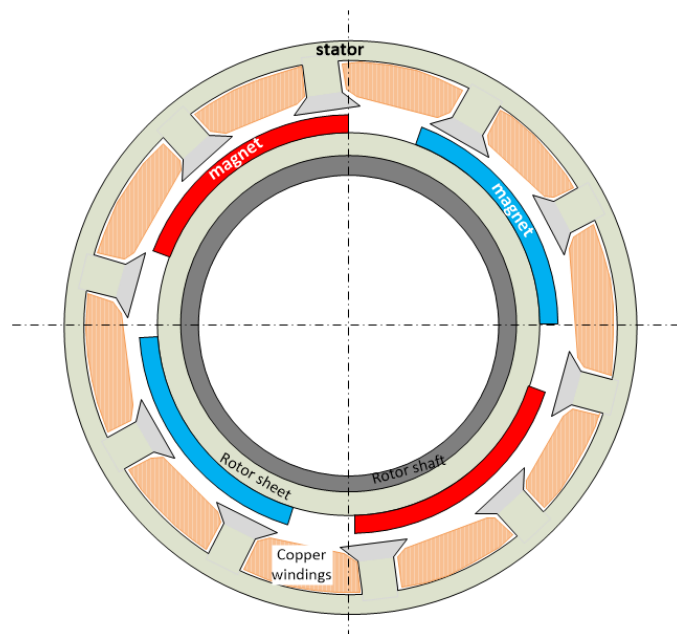


Figure 5: PMSM with inner rotor and surface mounted magnets

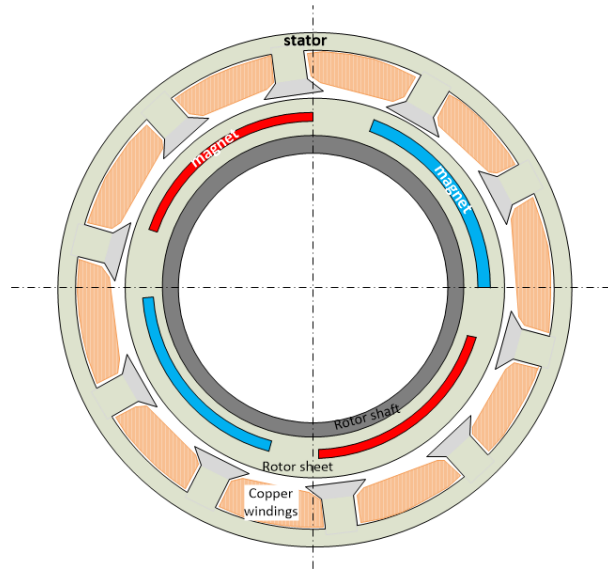


Figure 6: PMSM with inner rotor and buried magnets

Synchronous motors with permanent magnet often use construction with the outer rotor. On the next, Figure 7 is shown construction with the fixed-rotor (outer rotor).

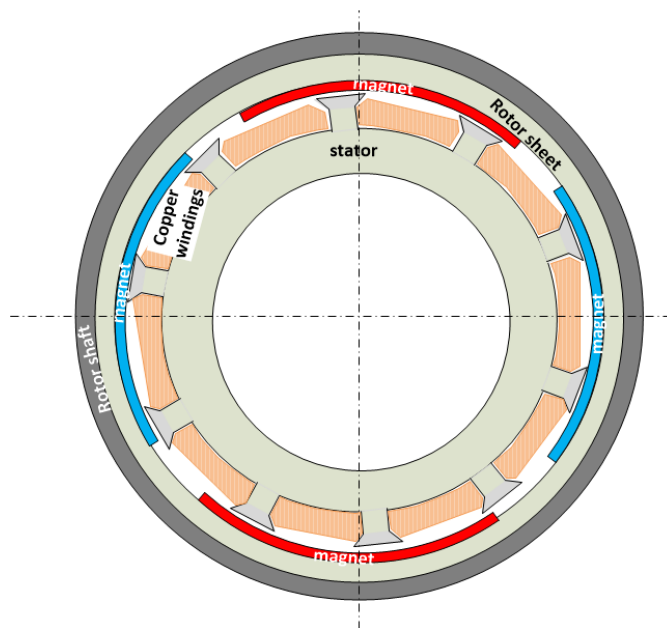


Figure 7: PMSM with fixed-rotor (outer)

Ironically, rotary stator with multi-phase (usually three-phase) winding is inside the construction, but the fixed-rotor with permanent magnets is located on outside. Permanent magnets can be installed in two different ways: *surface-mounted* and *buried permanent magnets*. The magnetic field in the air gap with fixed-rotor of the motor is influenced by the geometric dimensions and physical properties of permanent magnet, and by air gap size.

2.7 Mathematical description of synchronous motors

This section deals with the mathematical model of synchronous motor with permanent magnets. The following assumptions are made to derive the dynamic model [7]:

- The stator windings are balanced with sinusoidally distributed magneto-motive force;
- The inductance versus rotor position is sinusoidal;
- The saturation and parameter changes are neglected.

Each one of the 3 phases can be expressed by relation:

$$\begin{aligned} u_a &= R_S i_a + \frac{d\psi_a}{dt}, \\ u_b &= R_S i_b + \frac{d\psi_b}{dt}, \\ u_c &= R_S i_c + \frac{d\psi_c}{dt}. \end{aligned} \quad (2.14)$$

Where u_a , u_b and u_c symbolize the stator voltages, R_S is the resistance of a stator winding and i_a , i_b , i_c are the stator currents denoted in the rotor reference frame.

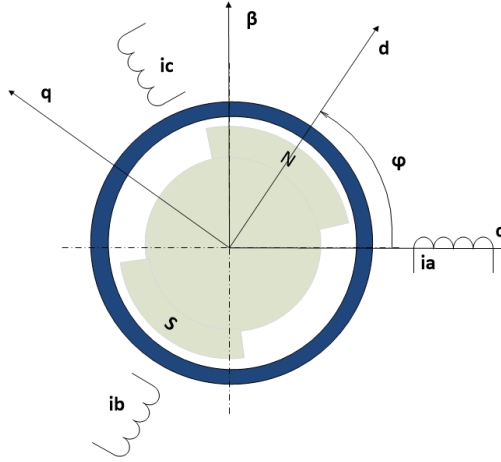


Figure 8: PMSM Reference Frames

The characteristics of the stator voltages are described by the following equations [6]:

$$\begin{aligned} u_q &= R_S i_d - \omega_e \psi_q + \frac{d\psi_d}{dt}, \\ u_d &= R_S i_q + \omega_e \psi_d + \frac{d\psi_q}{dt}. \end{aligned} \quad (2.15)$$

Stator's currents are marked as i_d and i_q ; ω_e is the electrical rotor speed and magnetic fluxes are ψ_d and ψ_q .

$$\begin{aligned}\psi_d &= L_d i_d + \psi_f, \\ \psi_q &= L_q i_q.\end{aligned}\tag{2.16}$$

Indexes d and q are new axes of relevant variables. The torque (T_q) is described in equation (2.17) [6].

$$T_q = \frac{3}{2} P_p [\psi_f i_q + i_d i_q (L_d - L_q)].\tag{2.17}$$

In the equation 2.17 variable P_p is the number of pole pairs. When it is exposed to a constant flux (which does not change the magnetic conductivity in the air gap), current i_d is close to zero and the electric motor torque is:

$$T_e = \frac{3}{2} P_p \psi_f i_q = K_t i_q.\tag{2.18}$$

Based on the equation of motion and fundamental rules for getting momentum relations:

$$\sum T_e = J \frac{d\omega_m}{dt},\tag{2.19}$$

$$T_e = T_L + K_{TL} \omega_m + J \frac{d\omega_m}{dt}.\tag{2.20}$$

T_L is the load torque, K_{TL} is the damping coefficient (for further adjustments will be neglected), ω_m is the mechanical speed of the motor, and J is the moment of inertia.

Computing all the equations from above (from 2.14 to 2.20), final equations that describe the PMSM are:

$$\frac{di_q}{dt} = \frac{u_d - R_S i_d + \omega_m L_q i_q}{L_d},\tag{2.21}$$

$$\frac{di_d}{dt} = \frac{u_q - R_S i_q + \omega_m L_d i_d - \omega_m \psi_f}{L_q},\tag{2.22}$$

$$\frac{d\omega_e}{dt} = \frac{P_p}{J} \left(\frac{3}{2} P_p [\psi_f i_q + (L_d - L_q) i_d i_q] - T_L \right),\tag{2.23}$$

$$\frac{d\phi}{dt} = \omega_e.\tag{2.24}$$

The electrical synchronous angular velocity is given as ω_e ($\omega_e = \omega_s$) and ϕ_e is angle position.

3 OPENMODELICA

Today, Modelica is one of the most promising modeling and simulation language.

OpenModelica is an open-source Modelica-based environment for modeling and simulating complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC).

OpenModelica deals with the physical modeling, object-oriented modeling and component-based modeling and simulation. Lot of components are already defined and mathematically described. It means there is no need for a new modeling and programming of these components, just to type their parameters and to start the simulation.

3.1 Asynchronous motor in OpenModelica

The asynchronous induction machine can be modeled in three different ways. First possibility is, like modeling in Matlab-Simulink environment, to make model by using the mathematical operators: integrators, gains, sum, Boolean step etc. Second way is to input already existing model of a three phase asynchronous induction machine with squirrel cage or with slip ring rotor. Path for these models in OpenModelica 1.9.0 is: *Modelica – Electrical – Machines – BasicMachines – AsynchronousInductionMachines - AIM_SquirrelCage* or *AIM_SlipRing*. Class parameterization modeling, the third and a the last possibility of making models and simulation in this environment, is by connecting and setting up electrical and mechanical components that OpenModelica offers. This type of modeling and making simulations is very important because there is a space to do changes in already existing classes.

3.1.1 Existing model of asynchronous induction machine with squirrel cage or ring rotor

This is the easiest way of simulating electrical machines in OpenModelica. To make full simulation, there is need to input following components on work space: *Model of a three phase asynchronous induction machine with squirrel cage or slip ring*, *Terminal box Y/D-connection*, *Multiphase sine voltage source*, *Star-connection*, *Ground node*, *ID-rotational component with inertia* and *load* which is presented with *step torque*. Already listed components from above are just for direct-on-line simulation example of asynchronous induction machine.

Second step is to type the parameters into component properties and to give them some instructions when and how to react. The basic component in simulations like this is Model of a three phase asynchronous induction machine with squirrel cage or slip ring, where is need to type *operational temperature of rotor resistance, number of phases and pole pairs, nominal frequency, inertia, inductances and stator resistances per phase*.

Star presents that the modeling scheme is multiphase. Parameter and variable “*m*” gives information about the number of phases. In this case, (Figure 9) AIMC is connected to the three-phase power system.

Terminal box is component that simulates connection of induction machine to the multiphase sine voltage source or to some controllers and drives. *Terminal box* offers a choice how to connect the induction machine, “*Y*” for a wye connection (typically use for high voltage) or “*D*” for a delta connection (typically use for low voltage).

Entering the details in multiphase *sine voltage source*, we set up the voltage level and the frequency for the asynchronous induction machine and our simulation scheme.

One of two mechanical components, which exists on following scheme (Figure 9), is *ID-rotational component*, contains the basic load inertia information. Second is rotational torque step that presents load for AIMC.

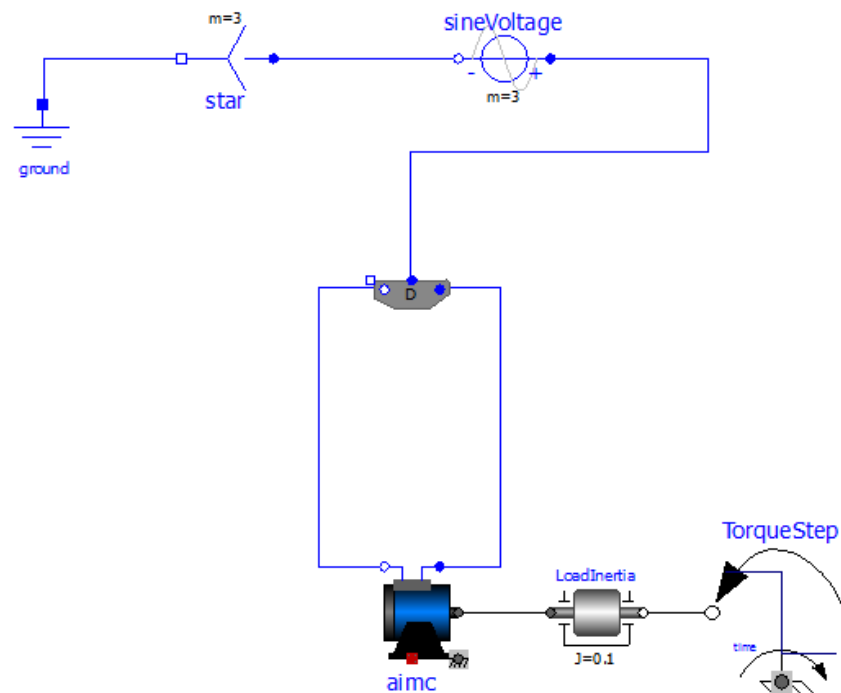


Figure 9: AIMC direct-on-line

3.1.2 Class parameterization modeling of AIMC

All advantages of OpenModelica, like multi-domain modeling, visual causal hierarchical component modeling, typed declarative equation-based textual language (defines all components of a model) and simulation that can be done in continuous time and discrete time, are shown in class parameterization modeling.

Modeling and simulation is important for understanding the complexity of systems, for optimizing systems and for system verification. Class parameterization modeling is typical way of physical modeling. Physical models have structure of the real systems but they use mathematical equations for expressing components parameters, the relationships in the system and between the components.

Class parameterization modeling of the asynchronous induction machine can be done in two different ways. The difference between the two class parameterization modeling is that in the first way class uses mathematical equations for current, voltage and flux linkage space vectors and in the second way class applies complex vectors for physical representation of the magnetic flux and the magnetic potential difference [8].

Both classes have the same mechanical components and parameters. The main difference in models is different description of the air gap and related components (stator core, inductor etc.). In the first case, *air gap* uses mathematical equations for current, voltage and flux and in the second case, *air gap* uses mathematical equations for magnetic reluctance.

A schematic for class parameterization modeling consists of connected components. Each component needs parameters to define the physical quantities. Figure 10 shows the class structure of the asynchronous induction machine without losses, where *air gap* (stator-fixed) uses mathematical equations for main electrical magnitudes.

The model of an ideal (without losses) asynchronous induction machine, on Figure 10, has no controller and it is connected direct-on-line to the three phase sin voltage source. *Negative torque*, quadratic dependent on angular velocity of flange, presents load on the AIMC.

Stator resistance per phase is indicated by R_s , L_{ssigma} is stator stray inductance in henries. *SpacePhasor* psychically transforms the three-phase values (voltages or currents) to space phase and zero sequence value. *AirGapS*, contains main field inductance L_m in henries, is air gap in stator-fixed coordinate system, using only mathematical equations. *SquirrelCage* contains rotor stray inductance per phase translated to stator (L_{rsigma} in henries) and rotor resistance translated to stator (R_r in ohms). Main AIMC inertia, *rotor inertia*, is presented with the *ID-rotational component*.

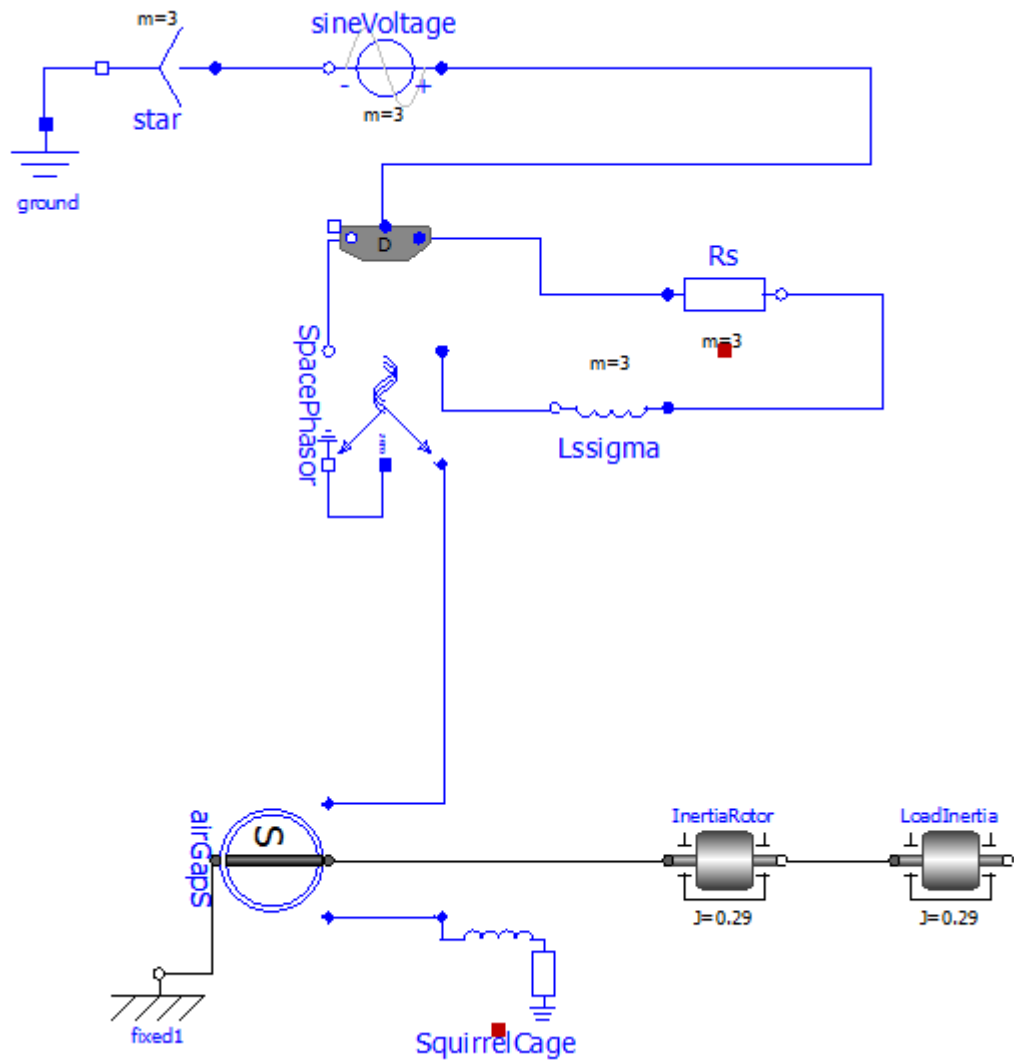


Figure 10: Class parameterization modeling of ideal AIMC direct-on-line

Big advantage of class parameterization modeling is that any magnitude from the scheme can be simulated and plotted.

Mathematical model of asynchronous induction machine with squirrel cage used for making the class in OpenModelica is *T-model*. It exists a few ways and mathematical descriptions how to modulate and simulate the AIMC and one of the most popular is exactly the *T-model* used on Figure 10.

Figure 10 doesn't show the *temperature losses*, *stray load losses*, *core* and *friction–mechanical losses* but it can be built into each component as well. All this specifications can be easily installed into OME model by adding the yellow boxes from the path: *Modelica – Electrical – Machines – BasicMachines – Losses*. Figure 11 shows AIMC with friction, stray load and core losses.

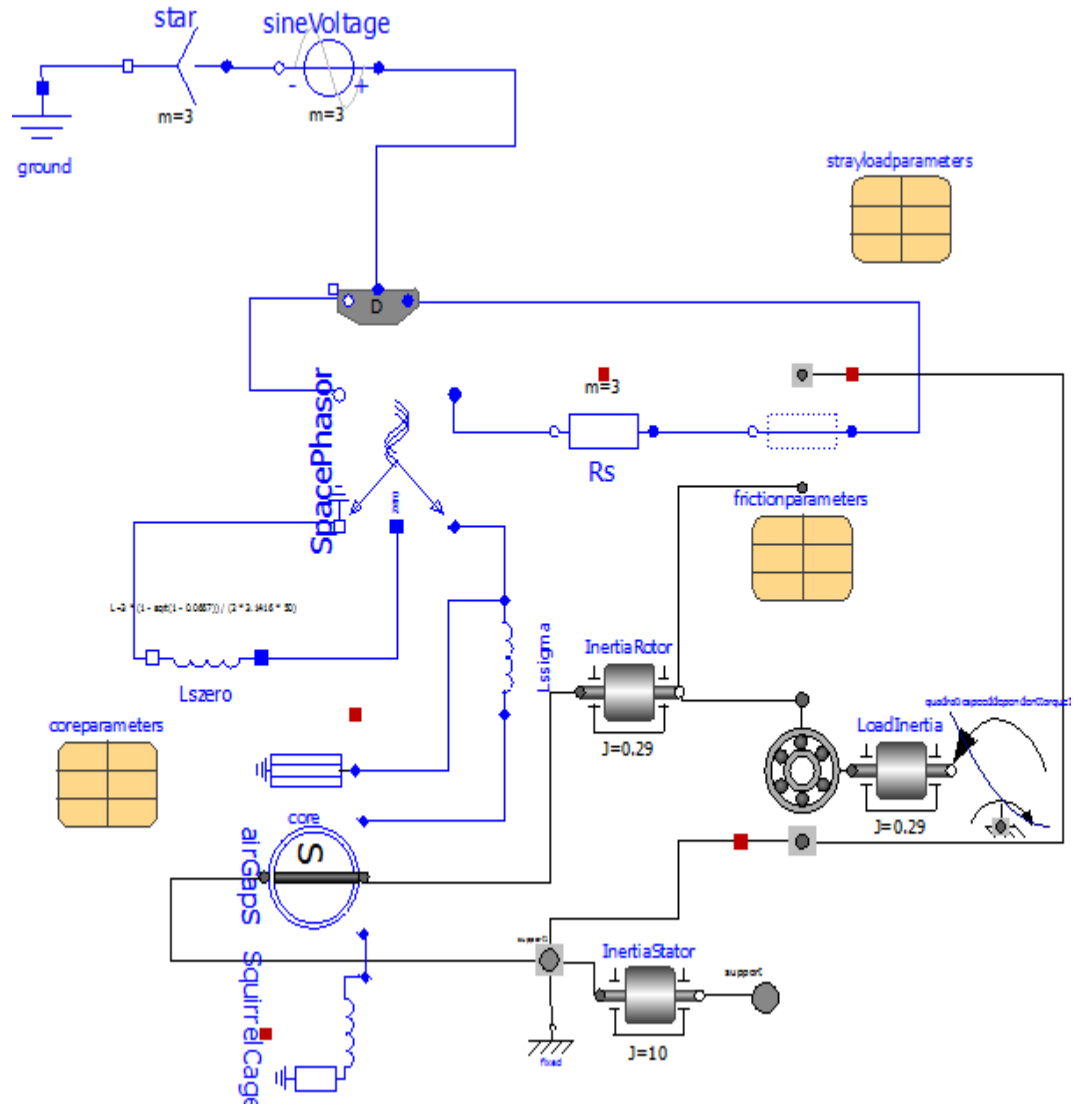


Figure 11: Class parameterization modeling of AIMC with losses, direct-on-line

Warm stator resistance per phase is indicated by R_s , L_{szero} is stator zero sequence inductance in henries. Parabolic arrow at the end of the load inertia is model of torque, quadratic dependent on angular velocity of flange (unit is Nm). Temperature losses are can be added to the model as well, but this wasn't the task of the bachelor's thesis. The other losses, such as: stray load, core and friction, are recognizable from the yellow boxes. Yellow boxes from the Figure 11 are actually the models in OME called "records". Records contain variable information, in this case losses.

By aligning the model of AIMC without losses (from Figure 10) with model of AIMC in Matlab-Simulink [9], it is possible to find the relationship between two models simulated in different modeling environments, to get mathematical equations and same simulation results. Two most important values, mechanical rotor speed (angular velocity in rad/s) and stator current in amps are presented on next figure.

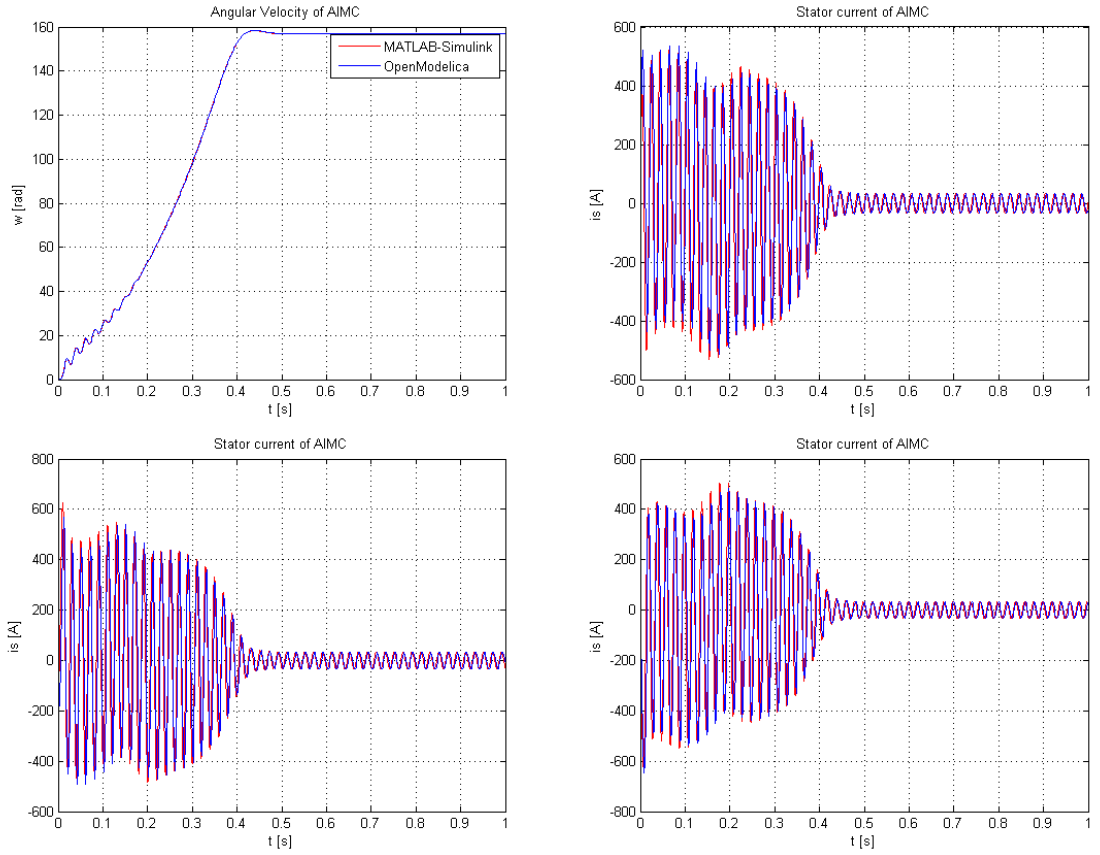


Figure 12: The mechanical rotor speed and the stator's currents of AIMC

Figure 12 presents the mechanical rotor speed and the stator's currents for models of an asynchronous induction machine with squirrel cage, where components have the same initialized values, simulated in the OpenModelica (blue color) and Matlab-Simulink (red color). Simulated mechanical rotor speeds in OME and Matlab-Simulink, above left on Figure 12, are overlapping. The same situation is with the stator's currents, and the graphs are overlapping. Both models, in OpenModelica and Matlab-Simulink are mathematically described as a T-model of AIMC. Because of that, there was no need to do any variable normalization and the values of AIMC are given in table on the next page.

Only visible small difference is in the stator's currents, shown above on Figure 12. Reasonable explanation for this difference in stator's currents plots is that Matlab-Simulink uses different numerical method with different numerical simulation step then compiler in OpenModelica. Default values for the asynchronous motor with squirrel cage used in simulation are given in table below.

This section verified the correctness and the accuracy of the AIMC model made in OpenModelica simulation environment.

Value description	Value	Unit
number of pole pairs p	2	-
nominal voltage per phase	100	V
nominal frequency f	50	Hz
nominal speed ω	1449	RPM
stator resistance R_S	0.03	Ω
rotor resistance R_R	0.04	Ω
stator zero sequence inductance L_{Szero}	0.32396	mH
stator stray inductance per phase L_{Ssigma}	0.32396	mH
rotor stray inductance translated to stator L_{Rsigma}	0.32396	mH
main field inductance per phase L_m	9.22531	mH
stator's moment of inertia J_S	0.29	kg m ²
rotor's moment of inertia J_R	0.29	kg m ²

Table 1: Variables of the AIMC

Model of asynchronous induction machine with squirrel cage (angular velocity and stator's currents) in OME is compared with the "Electromechanical Drives" [9] in Matlab-Simulink. This library is made by Ing. Petruška for his master's thesis.

3.2 Synchronous motor in OpenModelica

Modelica and OME environment contain the package with the models of a synchronous induction machines. Like the asynchronous motors in OME, synchronous induction machines can be modeled in three different ways.

Schemes and models of the *synchronous motors with permanent magnet, electric excited synchronous motor* and *synchronous reluctance motor* are already included in OpenModelica. Some typical examples of the synchronous motor (inverter, current source, voltage source, generator, load dump and rectifier) are given by OpenModelica as well: *Modelica – Electrical – Machines – Examples – SynchronousInductionMachine*.

This bachelor's thesis deals with the class parameterization modeling of the synchronous motors with permanent magnet and continues with the discussion about the similarities and differences of the parameterization modeling of the AIM with squirrel cage and SM with permanent magnet. The end of this chapter contains the plots of mechanical rotor speed and rotor current of models with and without losses.

3.2.1 Existing model of synchronous motor with permanent magnet

Adding already programmed models of synchronous motors is similar to the Simulink's Simscape library. These models can be used for testing system-level performances. By parameterizing each component of the models, system is ready for simulation and getting results (plotting). PMSM with inverter is shown on the Figure 13.

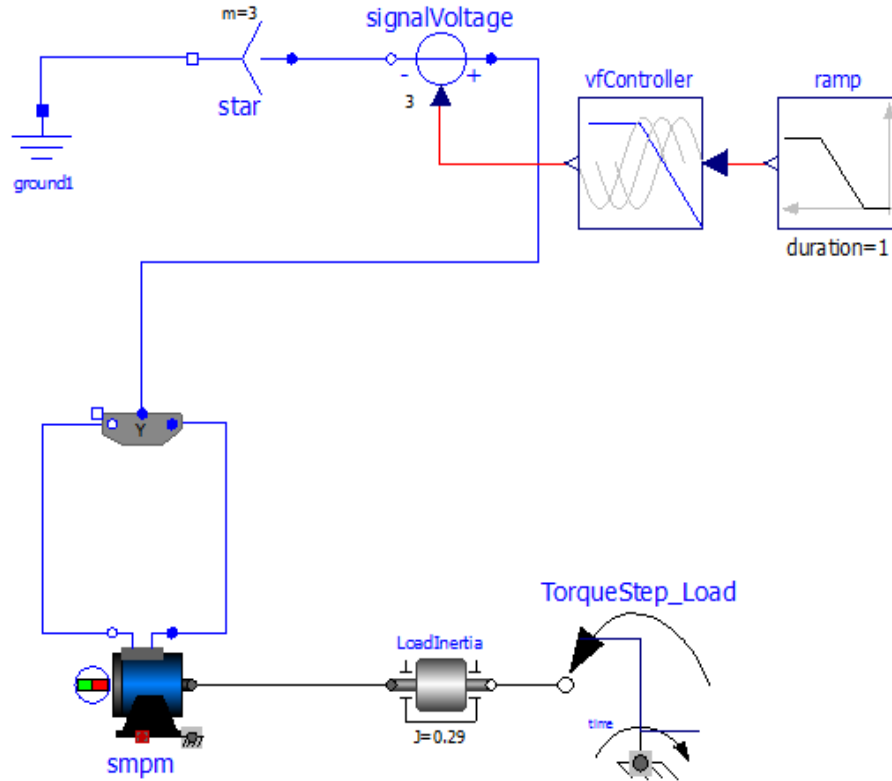


Figure 13: PMSM with inverter

3.2.2 Class parameterization modeling of SM with permanent magnet and losses

Ideal model (model without losses) of synchronous induction machine with permanent magnet and inverter is presented on the Figure 14.

In an induction machine, rotor is the component that converts the electrical power and delivers the mechanical power. In a case of PMSM rotor carries the permanent magnet and stator carries the conductors. Model of permanent magnet synchronous machine in OpenModelica environment exactly shows the physical existence of the permanent magnet, with north and south magnetic pole.

Table below shows default values used in simulation of the synchronous motor with permanent magnet and losses.

Value description	Value	Unit
number of pole pairs p	2	-
nominal voltage per phase	100	V
nominal frequency f	50	Hz
stator resistance R_S	0.03	Ω
rotor resistance R_R	0.04	Ω
stator stray inductance per phase L_{Szero}	0.15915	mH
stator stray inductance per phase L_{Ssigma}	0.15915	mH
stray inductance in d and q axis translated to stator	0.15915	mH
main field inductance per phase L_m	0.95493	mH
stator, rotor and load moment of inertia	0.29	kg m^2
reference friction losses at w_{Ref}	0.001	W

Table 2: Variables of the PMSM

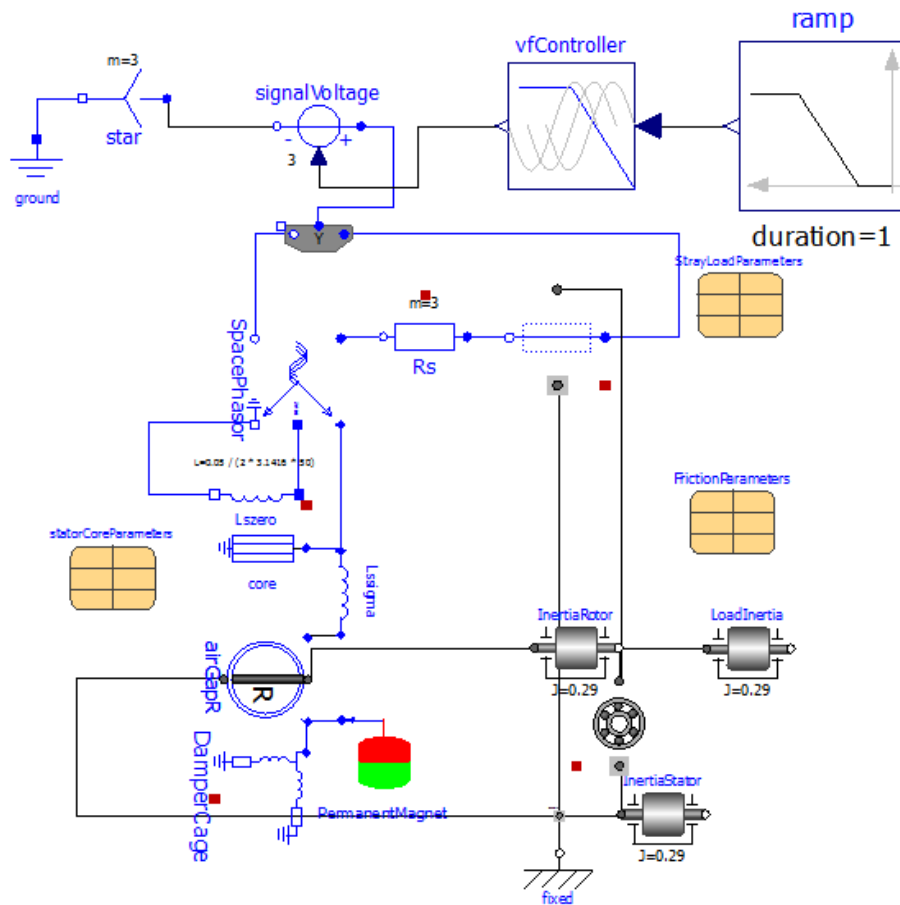


Figure 15: PMSM with inverter and losses

The model of SM with permanent magnet from the Figure 14 is similar to the model of AIM with squirrel cage. Warm stator resistance per phase, stator stray inductance per phase and rotor stray inductance per phase are included in both models. Only difference is that *air gap S* (stator-fixed) and squirrel cage is used for the simulation of ACIM and *air gap R* (rotor-fixed), *damper cage* and biggest difference is given in rotor - permanent magnet (as a rotor of the induction machine) is used for the simulation of SM with the permanent magnet. Figure 15 also includes some losses in the core of induction machine, stray load losses and mechanical (friction) losses.

These losses strongly affect mechanical rotor speed and rotor current. Figure 16 shows models in OME, where first is ideal model of PMSM without losses and second is non-ideal model of PMSM with losses. Both models have same parameters: nominal voltage per phase, frequency, pole pairs, rotor and stator resistances and inductances. Mechanical rotor speed and rotor current of the SM with losses are presented with blue color and with the red color are presented mechanical rotor speed and rotor current of the SM without friction, core and stray load losses.

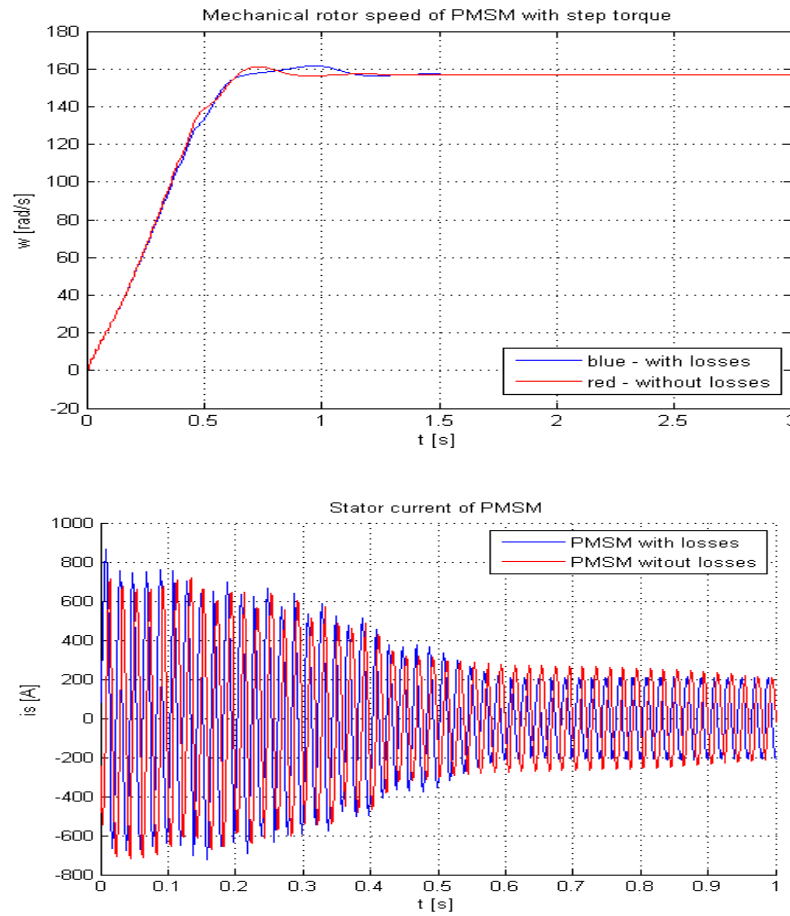


Figure 16: PMSM losses effect on angular velocity and stator current

Model of PMSM [9] in Matlab-Simulink is programmed for the small PMSM motors and it doesn't have *Damper Cage*. It means that the some variables (for model of PMSM in Matlab-Simulink) such as: *stray inductance* in *d*-axis and *q*-axis *per phase translated to stator*, *resistance translated to stator* and *equivalent excitation current* used for permanent magnet (in *amperes*) mathematical description are not defined. Model of PMSM simulated in OpenModelica, different modeling environment, supports and has all above mentioned variables and parameters. Extend of *equivalent excitation current* for permanent magnet; model from Matlab-Simulink defines variable called "*Mutual flux due to magnet*" in *volt-seconds*.

Equivalent excitation current for permanent magnet in OME is defined as:

$$I_e = \sqrt{2} \frac{Vs_{OpenCircuit}}{(2 * Pi * frequency * L_{md})}. \quad (3.1)$$

Mutual flux due to magnet has volt-seconds as a unit. From 3.1 equation current is defined as a volt per seconds (without inductance L_{md}). That's the relation between those two variables in OME and Matlab-Simulink.

Angular velocity and stator's currents of PMSM from OpenModelica and Matlab-Simulink have the same plots (Figure 17). This section verified the correctness and the accuracy of the PMSM model made in OpenModelica simulation environment.

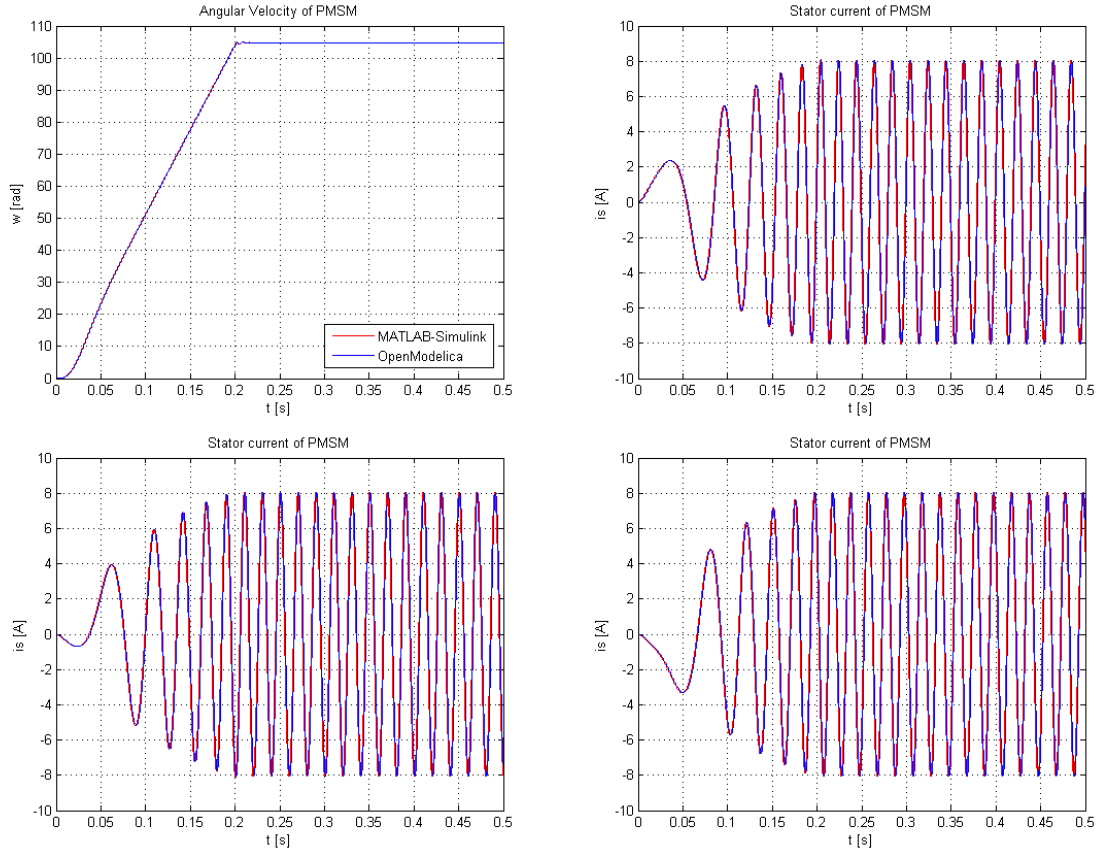


Figure 17: The mechanical rotor speed and the stator's currents of PMSM

3.2.3 PMSM with load

PMSM, as any other SM machine, runs at the same frequency or the rotation of the shaft is synchronized with the frequency of the supply current. Even if load is attached on the SM it should run at the synchronous speed. OpenModelica uses negative mechanical torque, from the path: *Modelica – Mechanics – Rotational – Sources*, to present load on machines. Figure 18 presents mechanical rotor speed (angular velocity) of the PMSM with step torque (start step time at $1s$) and detail of torque effect on SM. Default values of the PMSM are given in the table below (Table 3) the Figure 18.

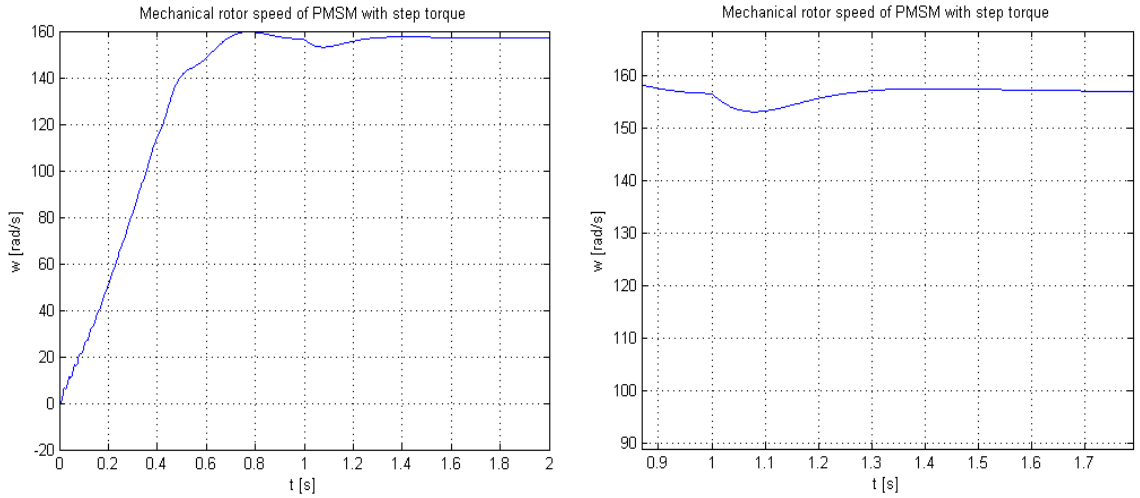


Figure 18: PMSM with load (direct-on-line), detail of step torque (right)

Value description	Value	Unit
number of pole pairs p	4	-
nominal voltage per phase	100	V
nominal frequency f	50	Hz
stator resistance R_S	0.03	Ω
rotor resistance R_R	0.04	Ω
stator stray inductance per phase L_{Szero}	0.15915	mH
stator stray inductance per phase L_{Ssigma}	0.15915	mH
stray inductance in d and q axis translated to stator	0.15915	mH
main field inductance per phase L_m	0.95493	mH
stator's moment of inertia J_S	0.29	kg m^2
rotor's moment of inertia J_R	0.29	kg m^2
load's moment of inertia J_L	0.29	kg m^2
Load M	50	Nm

Table 3: Variables of the PMSM with load

3.3 Sub models in OpenModelica

Classes in OpenModelica can be described as a sub models, models inside another model, which has *inputs/outputs* at the top level that can be connected to the sub models via connect or equations.

This bachelor's thesis used this advantage and presented AIMC and PMSM as sub models (classes) in bigger, more complex models with the regulators and electrical power systems. Setting the parameters of the AIMC and PMSM classes is by double clicking on the icon of the induction machine and it is better described in the chapter "3.6 Record and variable definition in OME".

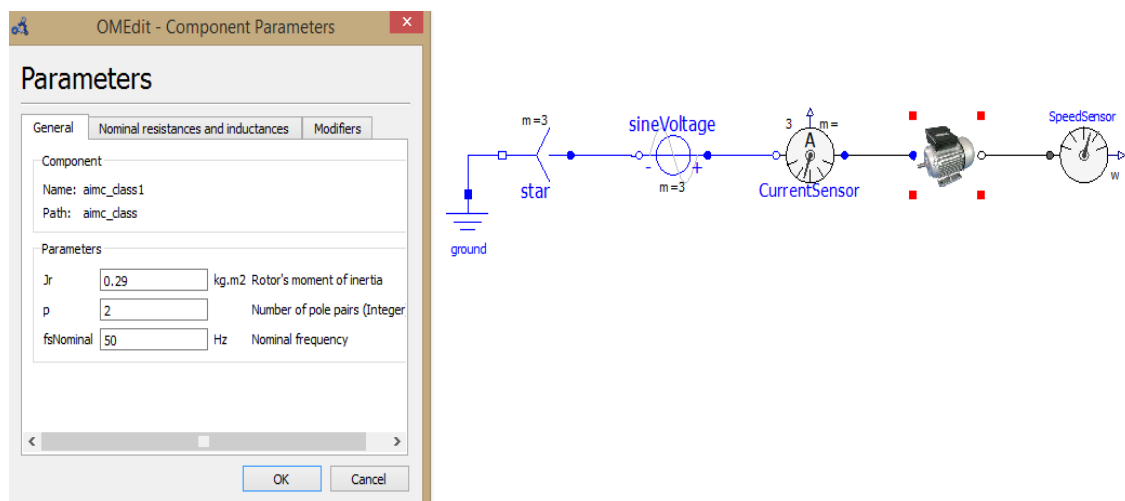


Figure 19: Sub Model of the AIMC with "Component Parameters"

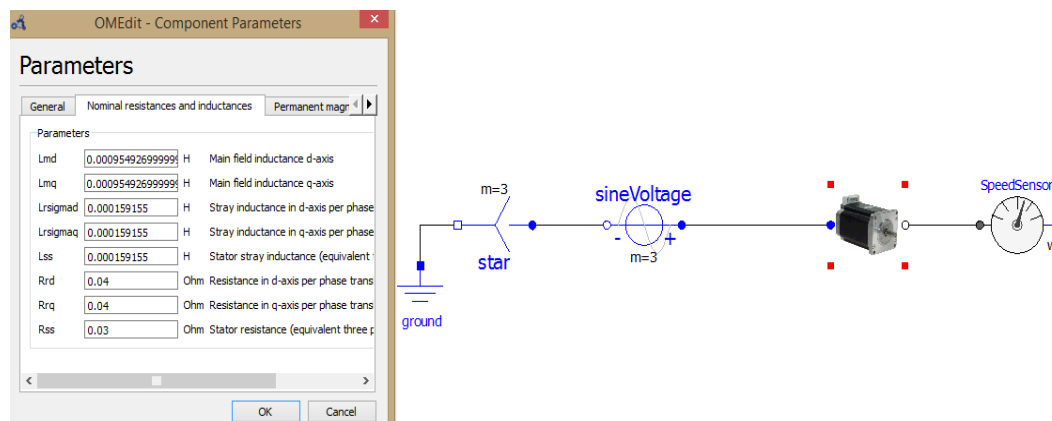


Figure 20: Sub Model of the PMSM with "Component Parameters"

Figures 19 and 20 from above, shows how the sub models of the asynchronous induction machine with squirrel cage and synchronous machine with permanent magnet are included and connected to the multiphase power system.

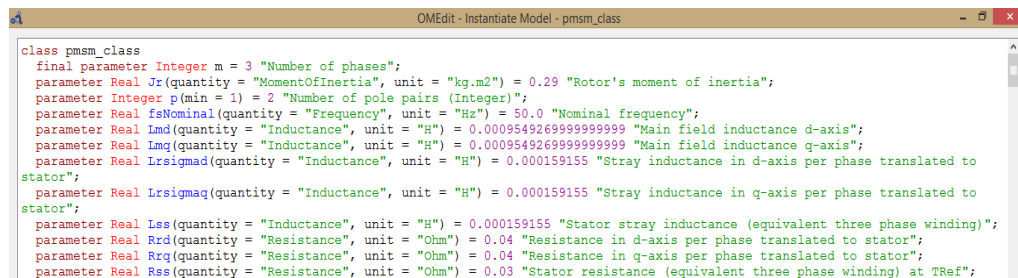
3.4 Instantiate model and text view

OpenModelica offers another way of defining models and provides notions of classes and objects. It is also called instances. About everything in OpenModelica is a class because the class concept is fundamental in OME.

OpenModelica “text view” shows the code that’s generated from the all blocks and elements with exact parameters of the model. Text view isn’t useful only for text reading, but unlike to the “*instantiate model*” that is useful only for code reading, text view provides doing some changings on model such as putting comments and quotes for some parameters, can change or add new variables, defines connection between the models and does everything else that can be done in Modelica but not in the OME environment, by simple adding the blocks and the components.

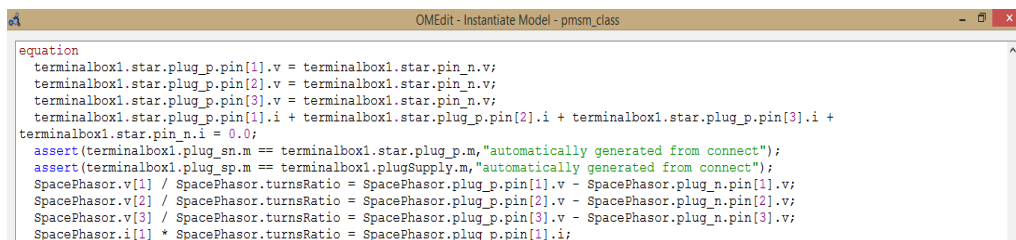
One of the biggest advantages of instantiate model in OME is clear-sightedness. Unlike to Matlab-Simulink, instantiate model in OME is understandable, readable and well commented where every value has a comment and unit.

Instantiate model of synchronous motor with permanent magnet is partly shown on the Figures 21 and 22.



```
class pmsm_class
  final parameter Integer m = 3 "Number of phases";
  parameter Real Jr(quantity = "MomentOfInertia", unit = "kg.m2") = 0.29 "Rotor's moment of inertia";
  parameter Integer p(min = 1) = 2 "Number of pole pairs (Integer)";
  parameter Real fsNominal(quantity = "Frequency", unit = "Hz") = 50.0 "Nominal frequency";
  parameter Real Lmd(quantity = "Inductance", unit = "H") = 0.0009549269999999999 "Main field inductance d-axis";
  parameter Real Lmq(quantity = "Inductance", unit = "H") = 0.0009549269999999999 "Main field inductance q-axis";
  parameter Real Lrsigmad(quantity = "Inductance", unit = "H") = 0.000159155 "Stray inductance in d-axis per phase translated to stator";
  parameter Real Lrsigmaq(quantity = "Inductance", unit = "H") = 0.000159155 "Stray inductance in q-axis per phase translated to stator";
  parameter Real Lss(quantity = "Inductance", unit = "H") = 0.000159155 "Stator stray inductance (equivalent three phase winding)";
  parameter Real Rrd(quantity = "Resistance", unit = "Ohm") = 0.04 "Resistance in d-axis per phase translated to stator";
  parameter Real Rrq(quantity = "Resistance", unit = "Ohm") = 0.04 "Resistance in q-axis per phase translated to stator";
  parameter Real Rss(quantity = "Resistance", unit = "Ohm") = 0.03 "Stator resistance (equivalent three phase winding) at TRef";
```

Figure 21: Definition of parameters in the instantiate model



```
equation
  terminalbox1.starplug_p.pin[1].v = terminalbox1.star.pin_n.v;
  terminalbox1.starplug_p.pin[2].v = terminalbox1.star.pin_n.v;
  terminalbox1.starplug_p.pin[3].v = terminalbox1.star.pin_n.v;
  terminalbox1.starplug_p.pin[1].i + terminalbox1.starplug_p.pin[2].i + terminalbox1.starplug_p.pin[3].i +
  terminalbox1.star.pin_n.i = 0.0;
  assert(terminalbox1.star.sn.m == terminalbox1.starplug_p.m,"automatically generated from connect");
  assert(terminalbox1.star.sp.m == terminalbox1.starplug_supply.m,"automatically generated from connect");
  SpacePhasor.v[1] / SpacePhasor.turnsRatio = SpacePhasor.starplug_p.pin[1].v - SpacePhasor.starplug_n.pin[1].v;
  SpacePhasor.v[2] / SpacePhasor.turnsRatio = SpacePhasor.starplug_p.pin[2].v - SpacePhasor.starplug_n.pin[2].v;
  SpacePhasor.v[3] / SpacePhasor.turnsRatio = SpacePhasor.starplug_p.pin[3].v - SpacePhasor.starplug_n.pin[3].v;
  SpacePhasor.i[1] * SpacePhasor.turnsRatio = SpacePhasor.starplug_p.pin[1].i;
```

Figure 22: Equations in the instantiate model

For better clear-sightedness text viewer in OpenModelica uses colors. With black color is marked *text* and the *names of the variables* used in instantiate model, brown is used for *keywords* (“parameter” refer to the Figure 21), red color shows the *type of the value* (“Real” refer to the Figure 21), blue means the *function*, *numbers* use purple color and for *quotes* and *comments* is used green color.

3.5 AIMC and PMSM documentation in OpenModelica

Big advantage of using OpenModelica is in making documentation of the classes and models. By clicking the button “*Documentation View*”, model description and documentation browser pop-up at the right window.

To make documentation for the new models, programmer should use instant model window in OME. Documentation definition starts at the very bottom of the instant model. By using simple HTML coding for making model documentation, OpenModelica avoids many difficulties and complications.

For both models in this bachelor’s thesis, asynchronous induction machine with squirrel cage and permanent magnet synchronous machine, documentation was implemented. Documentation browser shows the main information about AIMC and PMSM, variables and magnitude description used in model and information about author of these classes and models.

3.6 Record and variable definition in OME

The extension file name of everything, classes, models, block, function, connector etc. that is done in OpenModelica is “.mo”. The same is with the record of models.

Like is already mentioned, records are recognizable with “yellow boxes”, table that contains every parameter that is defined in OME tab “*Text View*”. For this bachelor’s thesis, already programmed record blocks and given by Modelica and they are used for the entering losses of the induction machines.

All the functionality that record blocks can do can be implemented into the class. This big advantage is used for defining the values of AIMC’s and PMSM’s parameters. To avoid some simulation errors or warnings the default values of variables are already given. User can change these values and set parameters of the induction machine simply by double-clicking on the AIMC or PMSM class.

4 FUNCTIONAL MOCK-UP INTERFACE – FMI

Functional Mock-up Interface - FMI is a standardized interface that supports both model exchange and co-simulation of models. In order to fulfill its mission, FMI uses a combination of *xml-files* and compiled *C-code*.

Main task of the FMI is to compile the FMU (functional mock-up unit). Each FMU model is actually a *zip-file* that inside itself contains compiled *xml-file*, where are information about parameters, variables and symbols used in model, and *C-code* which is used for mathematical derivations and getting the simulation results.

This bachelor's thesis uses FMI standard for translating models from OpenModelica, through JModelica (used for compilation of the FMU) to Matlab-Simulink.

4.1 JModelica used for the compilation of the FMU

First step to fulfill the quest and to make software that loads models from environment OpenModelica to Matlab-Simulink was to correctly create a functional mock-up unit of AIMC and PMSM with inputs and outputs. Idea was to create only the class model of induction machine without regulator or power supply and from this body to export the FMU. Creating the inputs is important because it allows to create any regulator in Matlab-Simulink and to connect it to the FMU of models made in OpenModelica environment.

OpenModelica environment provides us a solution that can export FMU from dynamics models, *mo-files*. Unfortunately, OpenModelica has a problem while compiles FMU-s of dynamics systems and they cannot be simulated in other modeling and simulation environments, but in OME. JModelica gives a way out.

JModelica, like OpenModelica modeling and simulation programming language, is a Modelica-based open source platform. It works with *mo-file* models, compiles those models to FMU, JMU and FMUX, loads models, changes the model parameters and accomplishes many other tasks.

The following steps are used for compilation of AIMC and PMSM models (*mo-files*) to the FMU in the JModelica, *pylab-bat* (IPython shell which also loads the numeric computation environment PyLab) [11]:

```
# Change the file directory  
cd 'C:\...\my_location_full_path\my_folder'
```

```

# Import the compiler function
from pymodelica import compile_fmu
# Specify aimc_class model and model file (for example aimc_class.mo)
model_name = 'aimc_class'
mo_file = 'aimc_class.mo'
# Compile the aimc_class.mo model
fmu_file = compile_fmu(model_name, mo_file)

```

Example from above is used for *aimc_class.mo* model of asynchronous induction machine with squirrel cage rotor, made in OpenModelica environment. The same had been done with the *pmsm_class.mo* model of permanent magnet synchronous machine.

Compiled AIMC FMU contains “*modelDescription.xml*”-file with the information about variables used in dynamic model, folder “*binaries\win32*” includes *aimc_class.dll-file* and folder “*sources*” includes *aimc_class.c-file*. The difference between JModelica FMUs and OpenModelica FMUs is that JModelica creates only one *dll-file* and only one source file. OpenModelica compiles more files, some mathematical, some system and model files. All of them are used for correct loading of the functional mock-up unit. This is a big disadvantage of OpenModelica. Because of that, FMU export of some complex dynamics system is faulty and JModelica appears as a better open-source solution for FMU compilation of AIMC and PMSM.

4.2 FMU Toolbox

Matlab-Simulink can’t export or import an *FMU-file*, but it offers solutions to make this real. Mission was to create a toolbox that can exchange models from Modelica to Matlab-Simulink using the functional mock-up interface (FMI) open standard. The main task of this toolbox was to load already exported functional mock-up unit of AIMC and PMSM into the Matlab-Simulink. They are a few ways how to work with FMU-files in Matlab-Simulink.

First option and idea was to use Matlab-Simulink *S-Function Builder*. The *S-Function Builder* enables the specification of the attributes of an *S-Function*. It automatically allows defining some inputs, outputs and relations between them. At first point, this was a visible solution, because *S-Function Builder* also offers importing and including some *C-code* files and the task of loading specific functional mock-up unit (AIMC or PMSM) would be possible. The problem was that Modelica uses “*gcc*” compiler and Matlab-Simulink under the Windows operating system doesn’t support above mentioned compiler. This final year project stuck at the linker error. Way out wasn’t possible and this solution was no longer in circulation.

Next idea was to use *input* and *output system files* of OpenModelica and Matlab-Simulink. In this case there is no need for exporting and importing the FMU. Both models, in OME and Simulink, exchanges simulation results by using files for reading and writing. The biggest disadvantage of this solution was the simulation speed. This solution offers the slowest and it consumes the most time and because this it wasn't satisfactory.

The best solution was to use the *QTronic's FMU SDK*, a free software development kit. Reason why to work with this software was that this program allows model exchange by using the functional mock-up interface standard.

Bc. Jan Glos already did research on this software development kit. His idea was to create dll-file from C-code of FMU and to use "*modelDescription.xml*"-file where are stored information about model variables. Software called "*FMU Toolbox*" [10] allows loading any functional mock-up unit into Matlab-Simulink. This is the biggest advantage of this software, but for this bachelor's thesis only AIMC and PMSM functional mock-up units are loaded into Matlab-Simulink. It also has a function to simulate any magnitude defined I the *xml-file*. FMU Toolbox does not require building outputs in the model. It offers option to simulate any variable on the Matlab-Simulink's scope by marking right check boxes. FMU Toolbox works with *32-bit* and *64-bit dll-files*, which is one of the biggest advantages of this software kit.

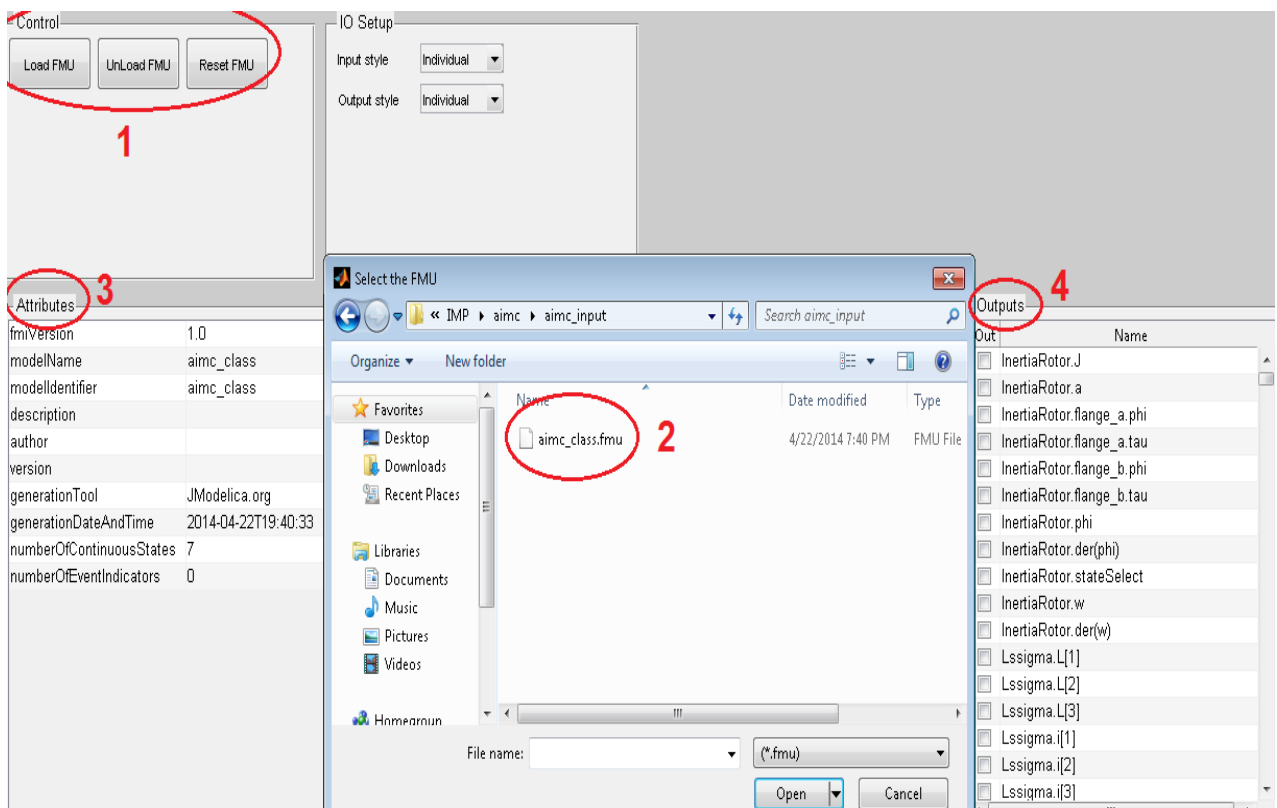


Figure 23: FMU Toolbox

1. Control

Control panel from the right top corner is the most important for FMU Toolbox. Users can load the FMU, Unload the FMU (changing the FMU doesn't require resetting of the FMU Toolbox or Matlab) and resetting already loaded FMU.

2. FMU file

By clicking the "*Load FMU*" button from the Control panel, users can select the path for the fmu-file and open it in FMU Toolbox, in Matlab-Simulink.

3. Attributes

All important information, such as the name of the loaded FMU, platform used for compilation of the FMU (in this case JModelica.org), generation time and date and many else, are given in the Attributes panel.

4. Outputs

In outputs section users can select which output to plot. If some outputs are not defined before FMU compilation, it doesn't make any problem to plot them. By simply clicking on the check boxes, Output panel makes the connections from FMU block to the scope in Matlab-Simulink of the signals.

One important option that FMU Toolbox doesn't provide is changing the parameters of asynchronous induction machine with squirrel cage in the fmu-file and permanent magnet synchronous machine in the *fmu-file*. It was necessitated to invent new software that does this function separately from the FMU Toolbox.

4.3 Parameterization Definer

"*Parameterization Definer*" gives an answer on problem mentioned in the previous chapter. It is a software developer kit that changes FMU parameters of the induction machines. Basically, it does some changings in the "modelDescription.xml"-file and redefine variables of the AIMC or PMSM.

Parameterization Definer is a simple script written in Matlab as an *m-file*. Functions of this software developer kit can be separated in the 7 groups:

- Rename *fmu* to *zip*;
- Extract the *zip-file*;
- Convert *xml-file* to *txt-file*;
- Import the new variables of the induction machine;
- Zip all files;
- Rename *zip* to *fmu* and call the destructor.

Every functional mock-up unit of some model is actually a zip-file. First step is to rename *fmu-file* to *zip-file* and to extract the contained files from this compressed. Matlab command “*movefile('file.xml', 'file.txt')*” can be used for renaming file and changing the file extension in the current folder (in this case *.xml* to *.txt*). The reason is that in *txt-file* is much easier to replace the text line then to work with the *xml-file*. At the very beginning of the Parameterization Definer script are definitions of the variables, where the user manually types the new values of the magnitudes. Parameterization Definer user can only work with this part. In the future, there is a plan to make GUI application and in that case, there will be not need to manually work with the script. Once all variables are set, files have to be compressed back to the *zip-file* and then renamed to *fmu-file*. At the end, destructor deletes some files, folders and closes some running processes used for proper functioning of Parameterization Definer software developer kit.

There are two Parameterization Definer scripts, one for the AIMC and second for the PMSM. Changing the variables of only two induction machines and no other model is big advantage that is used in this bachelor final project. In both already exported functional mock-up units programmer knows the exact variable and its position in the *xml-file* that has to be changed. Basically, script replaces old text line with the new one. This is how Parameterization Definer changes the values of the variables for AIMC and PMSM in FMU.

Setting values of variables for AIMC and PMSM in Parameterization Definer (in the Matlab script) are under “*Parameters of AIMC*” or “*Parameters of PMSM*”. Everything else is script-code and commands that users don’t change. All variables are commented in the script with the short description and unit. FMU-file of AIMC and PMSM are in the same folder as Parameterization Definer script.

This solution is simple and works correctly. It is fast and in less than one second Parameterization Definer software developer kit changes the all magnitudes of induction machine. Parameterization Definer is not standardized software and it can be used only for FMU of AIMC and PMSM made for this bachelor’s thesis. This is the biggest disadvantage of this script.

All variables of the FMU are given in the “*modelDescription.xml*”-file under the path: “*ModelVariables/ScalarVariable/Real*”. This definition of variables can be used for making the universal Matlab script that changes any variable of any model. This solution is visible in the future because Matlab can work with the *xml-files* and can change the “*start*” value of the functional mock-up unit.

5 CONCLUSION

Modelica programming language and environment OpenModelica is one of the most promising physical modeling, object-oriented modeling, component-based modeling and simulation and simulation language. This report introduced procedures of physical modeling and simulation of AC electrical machines and drives in OpenModelica, compares simulation results of the models in Matlab-Simulink and OpenModelica, talks about FMU and model exchange from OpenModelica to Matlab-Simulink. In contrast to the Matlab-Simulink, OpenModelica, an open-source Modelica-based language, is suitable for building losses into models of the asynchronous and synchronous motors. Instantiate model in OME is understandable and readable, due to its simple and clear-sightedness text viewer.

OpenModelica satisfies all expectations for modeling and simulating complex electrical systems as described in section 1 and 3 and successfully eliminates the difficulty of making models with stray load, friction, core and temperature losses. OpenModelica is a possible and commercially viable solution for this design problem.

FMU Toolbox and Parameterization Definer supports variables definition in Matlab and model exchange, among OME and Matlab-Simulink, co-simulation of dynamic models using a combination of xml-files and compiled C-code, due to Functional Mock-up Interface (FMI). Goal of bachelor's thesis was to make the classes of AIMC and PMSM without electrical power system in OpenModelica, to load, work with these classes in Matlab-Simulink and this task is successfully done.

In the future, some changes can be done for Parameterization Definer script, to make universal script for any FMU like it is already described in the last paragraph of the chapter "4.3 Parameterization Definer".

6 List of Figures

Figure 1: Stator [4]	3
Figure 2: Squirrel cage rotor [5]	3
Figure 3: Winding for deriving the mathematical model of ASM.....	5
Figure 4: Flux Density (B) versus Magnetizing Field (H) of Permanent Magnetic Materials.....	8
Figure 5: PMSM with inner rotor and surface mounted magnets.....	8
Figure 6: PMSM with inner rotor and buried magnets	9
Figure 7: PMSM with fixed-rotor (outer)	9
Figure 8: PMSM Reference Frames.....	10
Figure 9: AIMC direct-on-line.....	13
Figure 10: Class parameterization modeling of ideal AIMC direct-on-line	15
Figure 11: Class parameterization modeling of AIMC with losses, direct-on-line	16
Figure 12: The mechanical rotor speed and the stator's currents of AIMC.....	17
Figure 13: PMSM with inverter	19
Figure 14: Ideal PMSM with inverter	20
Figure 15: PMSM with inverter and losses.....	21
Figure 16: PMSM losses effect on angular velocity and stator current	22
Figure 17: The mechanical rotor speed and the stator's currents of PMSM.....	23
Figure 18: PMSM with load (direct-on-line), detail of step torque (right)	24
Figure 19: Sub Model of the AIMC with "Component Parameters"	25
Figure 20: Sub Model of the PMSM with "Component Parameters"	25
Figure 21: Definition of parameters in the instantiate model	26
Figure 22: Equations in the instantiate model.....	26
Figure 23: FMU Toolbox.....	30

7 Bibliography

- [1] Electrical machine. In *Wikipedia, The Free Encyclopedia* [online]. 2013, December 26. [Retrieved 00:43, December 31, 2013], from:
http://en.wikipedia.org/w/index.php?title=Electrical_machine&oldid=587734668
- [2] Tesla Constructs the First Induction Motor. In *Tesla Memorial Society of New York* [online]. 2005, March 22 [Retrieved 00:44, December 31, 2013], from:
<http://www.teslasociety.com/strasbourg.htm>
- [3] PAREKH, Rakesh. AC Induction Motor Fundamentals. In *Microchip Technology Inc.* [online]. 2003 [Retrieved 00:45, December 31, 2013], from:
<http://ww1.microchip.com/downloads/en/AppNotes/00887a.pdf>
- [4] Induction motor lecture Video. In *Electric Motor*. 2012, March 17 [Retrieved 00:47, December 31, 2013], from:
<http://dcacmotors.blogspot.cz/2009/04/induction-motor-lecture-vedio.html>
- [5] BALU, Sriram. Construction of squirrel cage induction motor explained. In *Bright Hub Engineering* [online] June 06, 2011 [Retrieved 20:49, January 01, 2014], from:
<http://www.brighthubengineering.com/diy-electronics-devices/43723-how-are-squirrel-cage-induction-motors-constructed/>
- [6] ENGST, C. *Object-Oriented Modelling and Real-Time Simulation of an Electric Vehicle in Modelica: Modeling of controlled synchronous induction machines with permanent magnets in Powertrain 2.1 library*. Munich, 2010. Master Thesis. Technical University of Munich, DLR & TUM EAL.
- [7] GANGADHARA RAO VENNA, S., S. VATTIKONDA and S. MANDARAPU. Mathematical modelling and Simulation of Permanent Magnet Synchronous Motor. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*. August 2013, Vol. 2 Issue 8, pages 7 [Retrieved 19:15, January 01, 2013]. ISSN (Online): 2278 – 8875.
- [8] KRAL, Christian and Anton HAUMER. *The New FundamentalWave Library for Modeling Rotating Electrical Three Phase Machines: Proceedings of the 8th Modelica Conference*. March 20-22, 2011, pages 10 [Retrieved 05:28, January 15, 2013]. AIT Austrian Institute of Technology GmbH Mobility Department, Electric Drive Technologies Giefinggasse 2, 1210 Vienna, Austria.
- [9] PETRUŠKA, L.: *Simulace řídicích struktur elektromechanických systémů*. Diplomová práce. FEKT VUT v Brně, 2010, 90s.

- [10] GLOS, J.. *FMU Toolbox* [software]. April, 2014. Available from:
<http://www.vutbr.cz>. System Requirements: Operating System Microsoft Windows 7 (32-bit), Matlab-Simulink, Microsoft Windows SDK, .NET Framework 4.
- [11] Modelon AB. *JModelica.org User's Guide 1.13I* [online]. Modelon AB, © 2014. March 26, 2014 [Retrieved 23:57, May 22, 2014]. Available from:
<http://www.jmodelica.org/page/6212>

8 List of abbreviations and symbols

OME	OpenModelica Editor
AIM	Asynchronous induction machine
AIMC	Asynchronous induction machine with squirrel cage
SM	Synchronous machine
PMSM	Permanent Magnet Synchronous Machine
AC	Alternating current
NdFeB	Neodymium-Boron Iron
SmCo	Samarium-Cobalt
N_s	Synchronous speed of the stator magnetic field [RPM]
f	Frequency [Hz]
P_p, p	Number of pole pairs [-]
N_B	Base speed for asynchronous induction motor [RPM]
U_s	Harmonic voltage [V]
u_a	Voltage per phase a [V]
u_b	Voltage per phase b [V]
u_c	Voltage per phase c [V]
\mathbf{u}_s^S	Stator voltage [V]
\mathbf{u}_r^R	Rotor voltage [V]
\mathbf{i}_s^S	Stator current [A]
\mathbf{i}_r^R	Rotor current [A]
R_s	Resistance of stator winding [Ohm]
R_r	Resistance of rotor winding [Ohm]
L_s	Inductance of stator winding [H]
L_r	Inductance of rotor winding [H]
L_m	Main field inductance [H]
T	SI unit for magnetic field strength or magnetic flux density
B	Flux Density [T]
H	Magnetizing Field [A/m]
i_a, i_b, i_c	Stator currents denoted in the rotor reference frame [A]
m	Number of phases [-]
R_s	Warm stator resistance per phase [Ohm]
L_{sigma}	Stator stray inductance per phase [H]
$L_{r\text{sigma}}$	Rotor stray inductance (equivalent three phase winding) and rotor stray inductance per phase translated to stator [H]
L_m	Main field inductance [H]
L_{zero}	Stator zero sequence inductance [H]
.txt	Text file
.xml	Extensible Markup Language
FMU, FMI	Functional Mock-up Unit, Functional Mock-up Interface
HTML	HyperText Markup Language

9 List of attachments

1. Attachment	AIMC folder	Sub model and class of AIMC and full model of AIMC connected to the 3-phase sin Voltage.
2. Attachment	PMSM folder	Sub model and class of PMSM and full model of PMSM connected to the 3-phase sin Voltage.
3. Attachment	AIMC_losses folder	Sub model and class of AIMC with losses and full model of AIMC with losses connected to the 3-phase sin Voltage.
4. Attachment	PMSM_losses folder	Sub model and class of PMSM with losses and full model of PMSM with losses connected to the 3-phase sin Voltage.
5. Attachment	Parameterization Definer for AIMC	Script used in Matlab for defining the variables of AIMC
6. Attachment	Parameterization Definer for PMSM	Script used in Matlab for defining the variables of PMSM
7. Attachment	aimc_class.fmu	FMU of AIMC compiled in JModelica
8. Attachment	pmsm_class.fmu	FMU of PMSM compiled in JModelica
9. Attachment	aimc_no_losses.mo	Model of a three phase AIMC direct-on-line in OpenModelica without losses.
10. Attachment	om_aimc.mo	Model of a three phase AIMC direct-on-line in OpenModelica with losses.
11. Attachment	ome_pmsm.mo	Model of a three phase PMSM with inverter in OpenModelica without losses.
12. Attachment	om_sm.mo	Model of a three phase PMSM with inverter and without losses in OpenModelica.
13. Attachment	losses_sm.mo	Model of a three phase PMSM with inverter and losses in OpenModelica.
14. Attachment	aimc_class.fmu	FMU-file of AIMC with inputs and outputs.
15. Attachment	pmsm_class.fmu	FMU-file of PMSM with inputs and outputs.